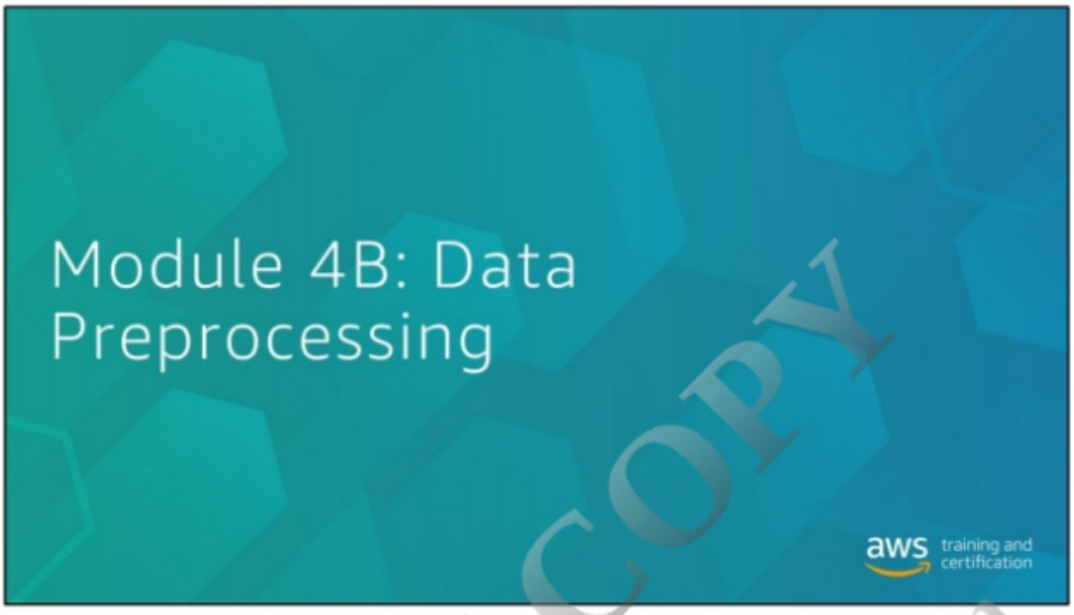
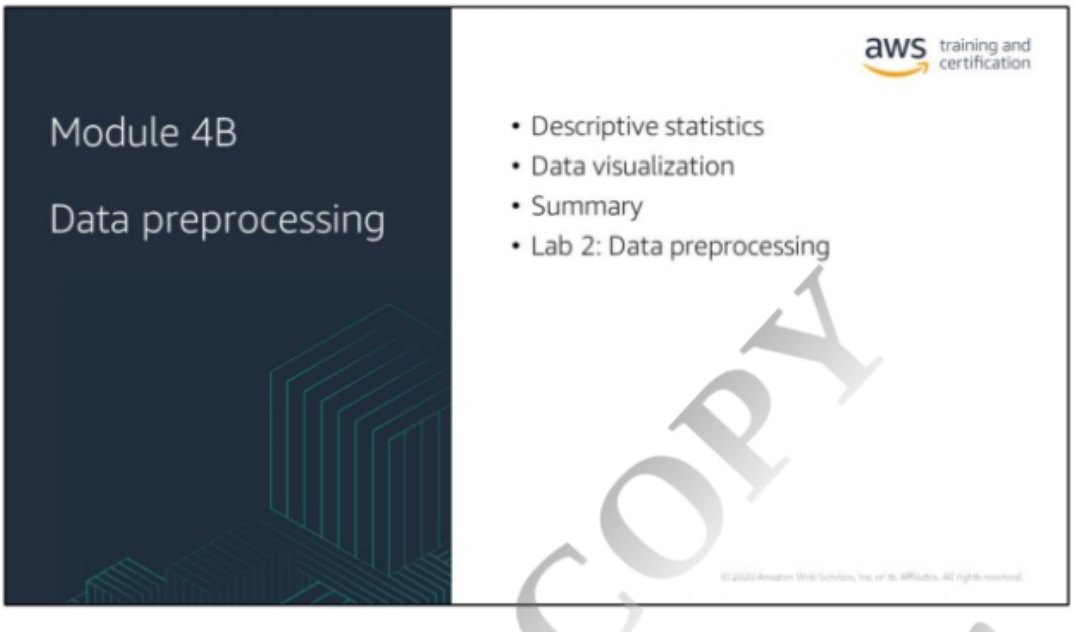




This is the second part of the data preprocessing module. Here we are going to continue discussing data preprocessing, before we talk about how you can visualize different features and stats. Like the other modules, we'll end this module with hands-on practice and some project work.

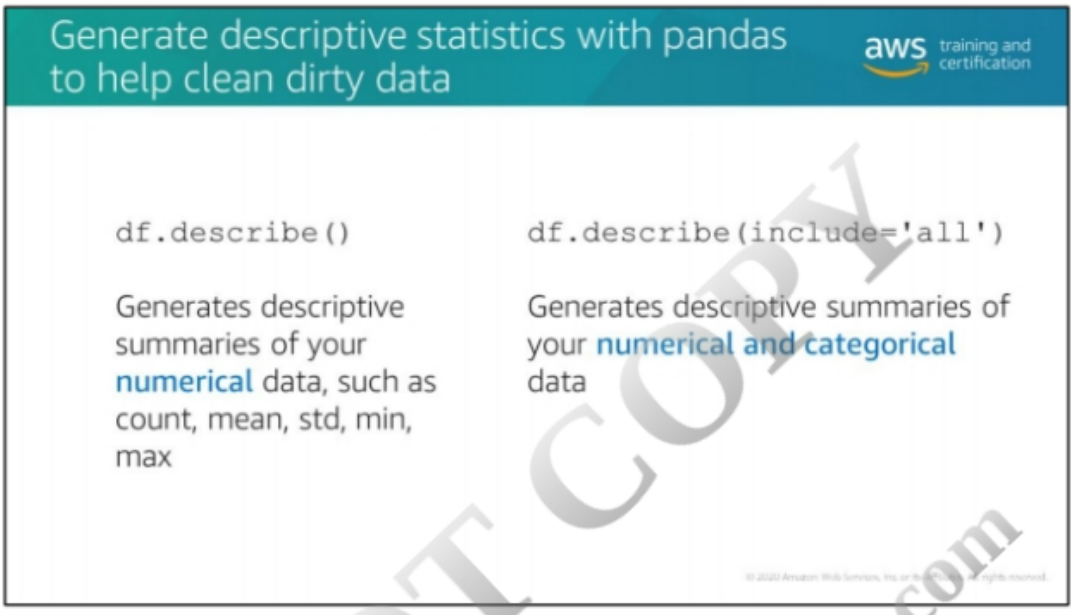
Outliers in the data

aws training and certification

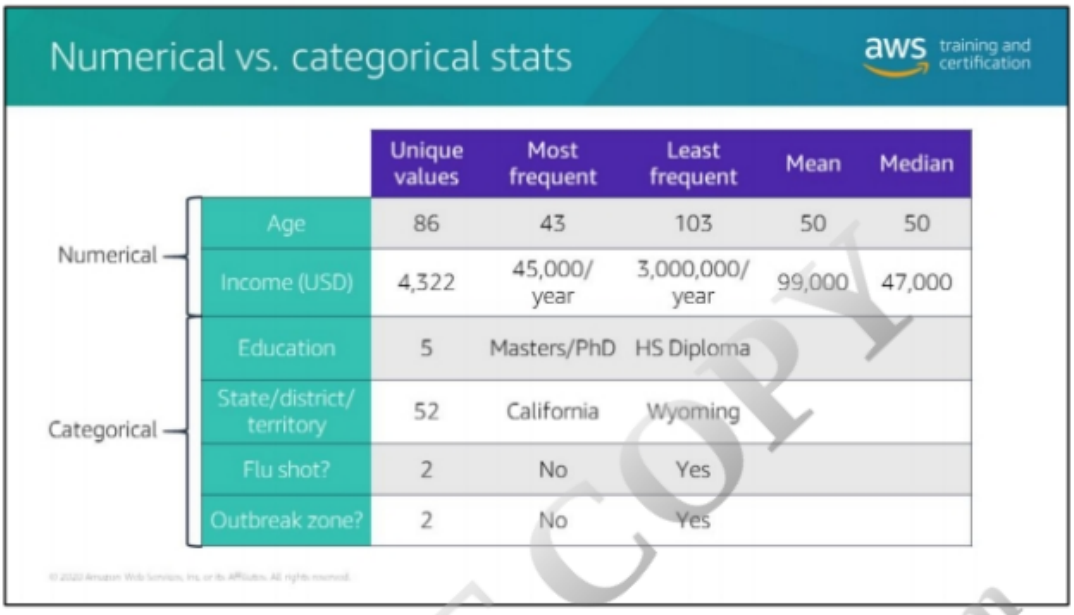
Participant ID	Age	Income	Education	State	Flu shot	Outbreak zone?
123456	39	45,000/ year	4 yr degree	New York	Yes	No
123457	23	18,000/ year	HS diploma	Minnesota	No	Yes
123458	78		Masters/PhD		Yes	Yes
123459	20	3,000,000 /year	HS diploma	California	No	No
123460	154	53,000 /year	Masters/PhD			
...	...	...	...	...	...	...

Outliers?

A common type of dirty data involves outliers. Outliers will influence your data and you need to determine if they're appropriate, or if they need to be changed or removed.



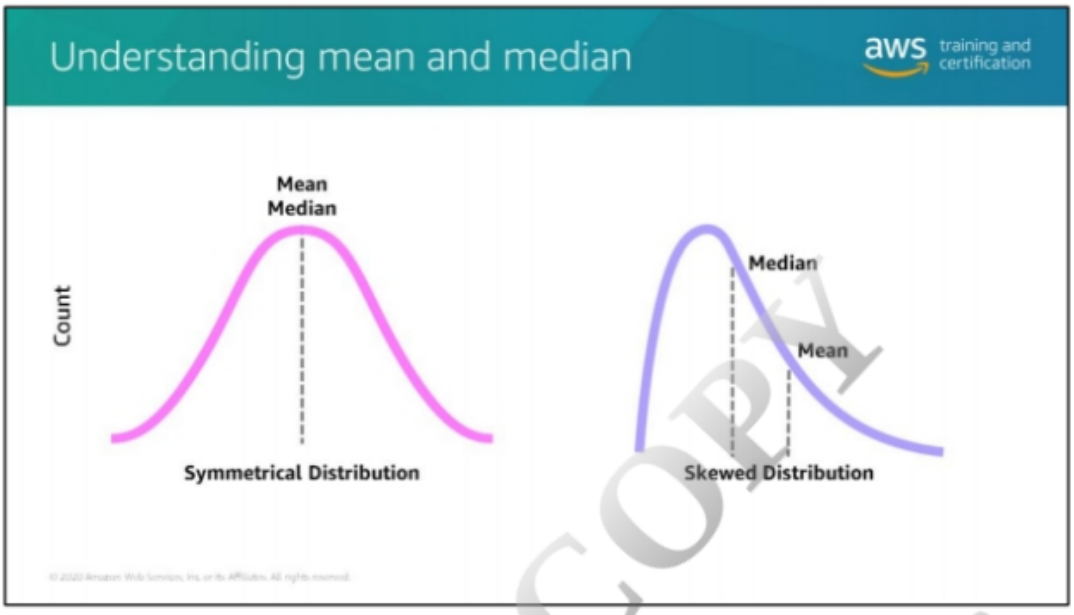
With Pandas, for instance, you can call the shape object to see your dataset's dimensions.



For *categorical attributes*, you can look at the frequency of attribute values in your dataset. That information is going to give you some idea about what is inside that categorical variable.

Specifically, for a target variable which is also of categorical type, you can look at the class distribution to see whether there is a class imbalance in your dataset. Imbalanced data can mark a disproportionate ratio of your classes—for instance, if your dataset is made up of credit card transactions, but only a tenth of a percent is labeled as *fraud*. In this case, your algorithm may not learn adequately enough to predict examples of credit card fraud.

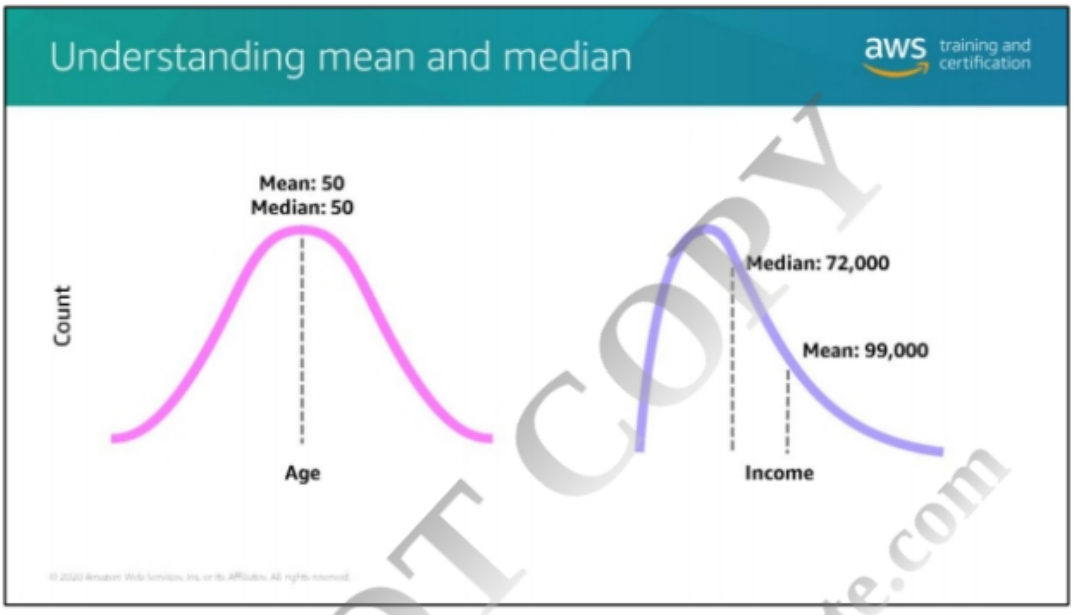
Note: The "State/District/Territory" column has 52 "unique values" because it includes Washington DC and Puerto Rico.



Mean and *median* are two different measures describing the extent to which your data is clustered around some value or position.

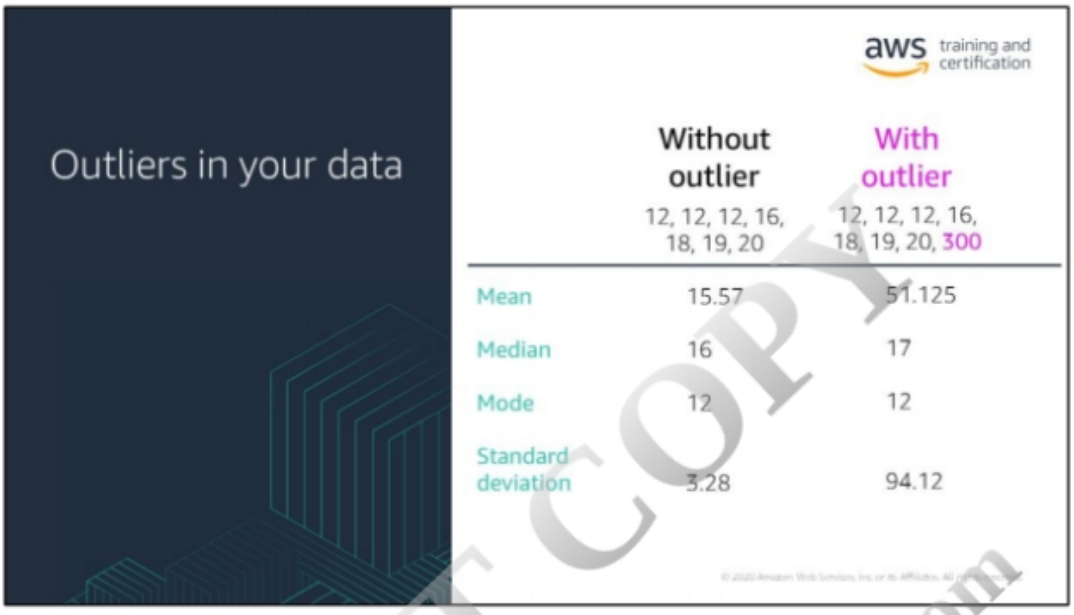
Mean can be a useful method for understanding your data when the data is symmetrical.

However, if your data is skewed or contains outliers, then median tends to provide the better metric for understanding your data as it relates to central tendency. For instance, if you have outliers with large values, the mean can be skewed one way and therefore not serve as an accurate representation of where your values are truly centered. Median doesn't get affected by outliers in the same way.



Your age data may be symmetrical because the age mean and median are so close, whereas income is skewed toward high incomes because a small number of very high incomes are dragging the mean up.

You may also need to clean your data based on any outliers that may exist. *Outliers* are points in your data set that lie at an abnormal distance from other values. They are not always something you want to clean up, because they can add richness to your dataset. But they can also make it harder to make accurate predictions, because they skew values away from the other more normal values related to that feature. Moreover, an outlier may also indicate that the data point actually belongs to another column.



Outliers can have a real impact on your dataset. Let's look at an example of two small datasets, one with outliers and the other without.

Look at how much different the mean and standard deviation are for the dataset with the outlier. Rather significant differences that could really impact your view of the overall central tendency and range of your data.

Is your outlier natural or artificial?

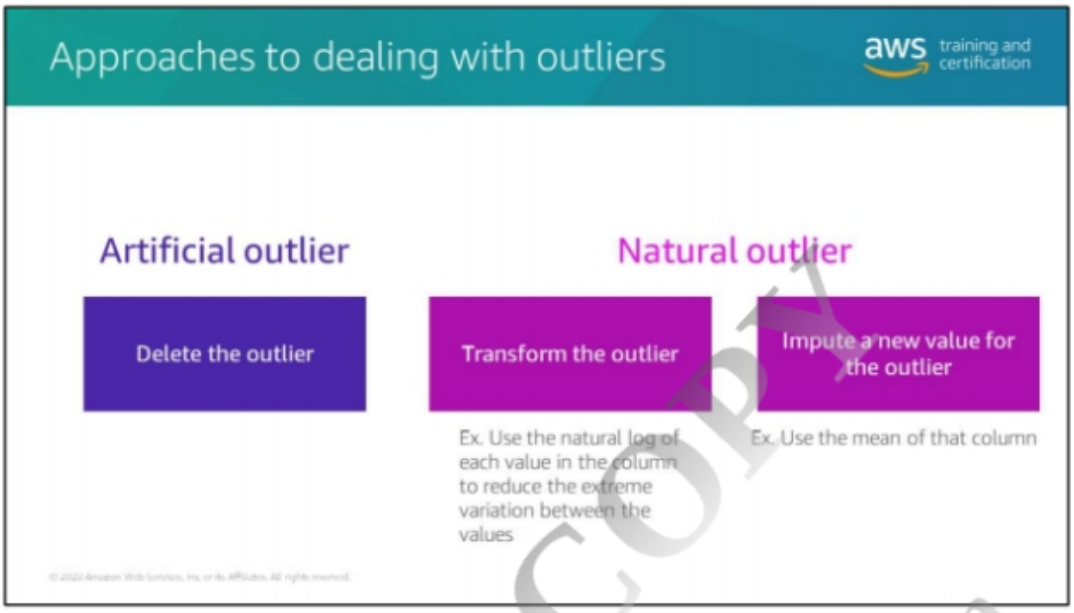
aws training and certification

Participant ID	Age	Income	Education	State	Flu shot	Outbreak zone?
123456	39	45,000/ year	4 yr degree	New York	Yes	No
123457	23	18,000/ year	HS diploma	Minnesota	No	Yes
123458	78		Masters/PhD		Yes	Yes
123459	20	3,000,000 /year	HS diploma	California	No	No
123460	154	53,000 /year	Masters/PhD			
...	...	...	...	...	...	...

Natural

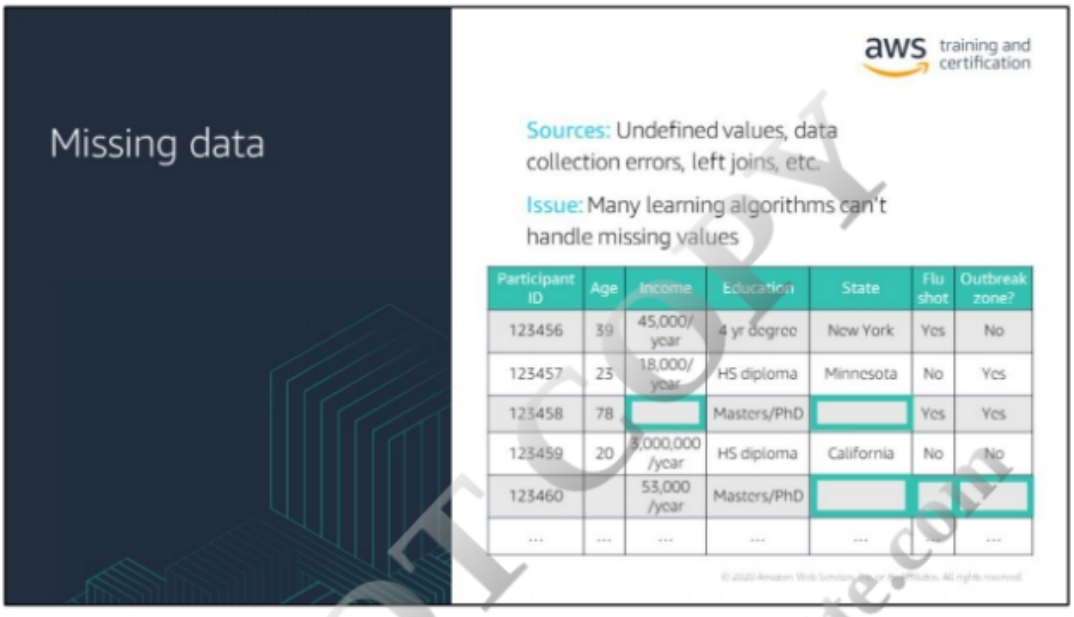
Artificial

Outliers can be caused by either natural or artificial factors. A natural outlier is not the result of some artificial error but instead is reflective of some truth in the data. For instance, you could have one home price that is several hundred thousand dollars more than the average home price in a dataset. But you can also have outliers that are indeed caused by artificial errors. Among other types of errors, these might include data entry errors that lead to the outlier.



The origin of your outlier will most likely inform how you deal with it during this data preprocessing phase of the pipeline, or possibly later during feature engineering. There are several different approaches to dealing with outliers. They include, but are not limited to:

- **Deleting the outlier:** This would be the approach to use especially if your outlier is based on an artificial error.
- **Transforming the outlier**: You could do this by taking the natural log of a value, which in turn would reduce the variation caused by the extreme outlier value and therefore the outlier's influence on the overall dataset.
- **Imputting a new value for the outlier**: You could use the mean of the feature, for instance, and **imput that value to replace the outlier value**. Again, this would be a good approach if the outlier was caused by artificial error.



In addition to outliers, you may also find that you have missing data. For example, some columns in your dataset may be missing data due to a data collection error, or perhaps data was not collected on a particular feature until well into the data collection process. Missing data can make it difficult to accurately interpret the relationship between the related feature and the target variable, so, regardless of how the data ended up being missed, it is important to deal with the issue.

Unfortunately, most ML algorithms cannot deal with missing values automatically. We have to use human intelligence to replace missing values with something meaningful and relevant to the problem.

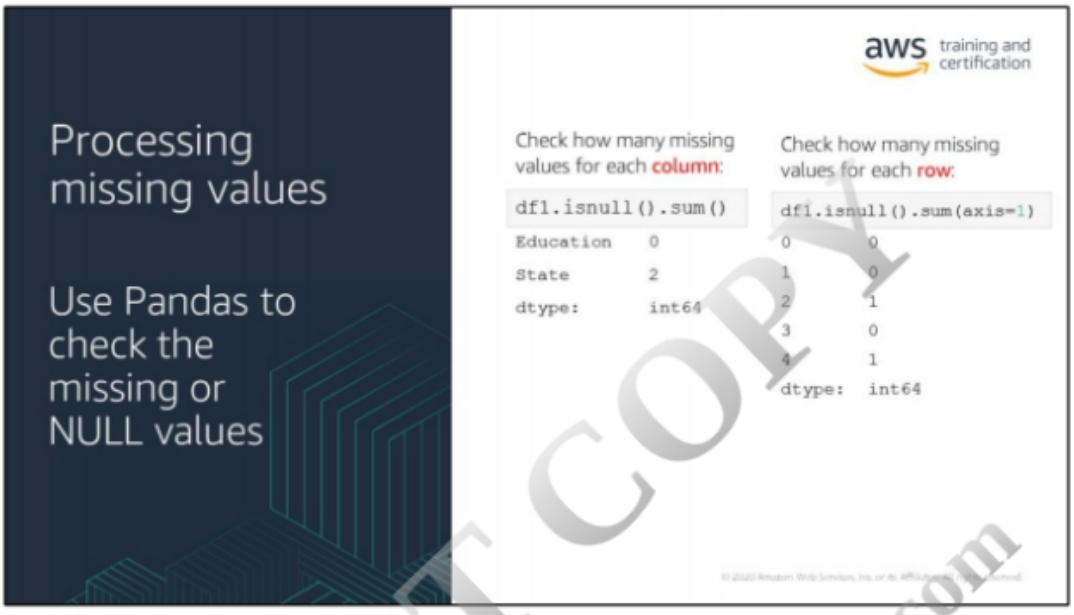
Missing data makes it hard to interpret a feature/target relationship

aws training and certification

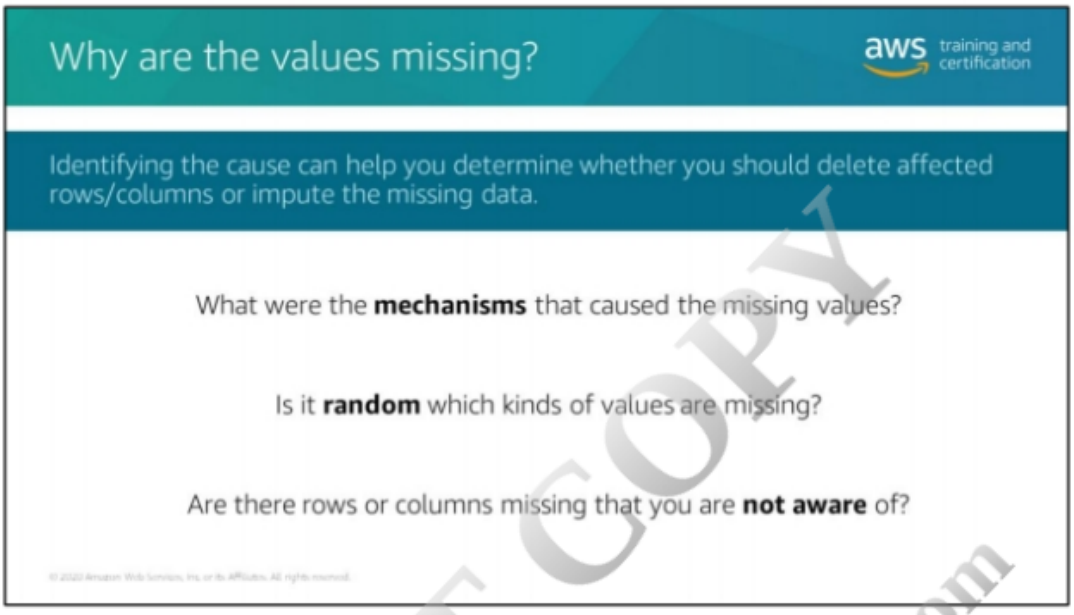
```
df1= df[['Education','State']]
```

Education	State
4 yr degree	New York
HS diploma	Minnesota
Masters/PhD	
HS diploma	California
Masters/PhD	
...	...

A few missing values might not be an issue, but if you're missing quite a few of the values in one column, you may find it very difficult to interpret the relationship between that feature and the target (value in that row that corresponds to what needs to be predicted by the model).



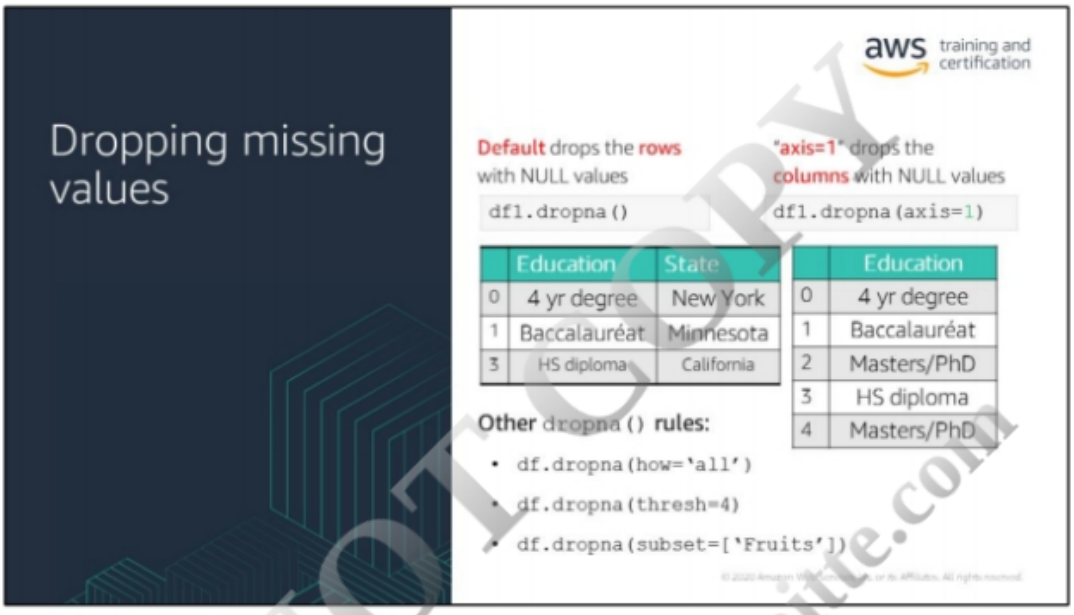
In Pandas there are some useful tools for us to identify which row is missing or which column contains a missing value (the default is by column). We can use the is_null function to look at each column and check for any missing values and then sum the total number of missing values up and/or give us a percentage of data missing in that particular column. We can do the same for each row and by using the keyword of "axis=1".



So how do you decide if you should drop or impute missing values? This question is answered in part by better understanding how those values came to be missing in the first place and how much data the missing values represent within your larger dataset.

For instance, if the missing values are quite randomly spread throughout your dataset, this could be due to a random failure in the data capture mechanism and doesn't represent a larger portion of its respective row or column because it's largely random where the data is missing. In this case imputation is most likely the better option.

On the other hand, if you have a column or row that has a large percentage of missing values, dropping the entire row or column would be preferred over imputation. For example, a majority of respondents to a survey may have felt uncomfortable answering certain questions, filling that column with missing values.



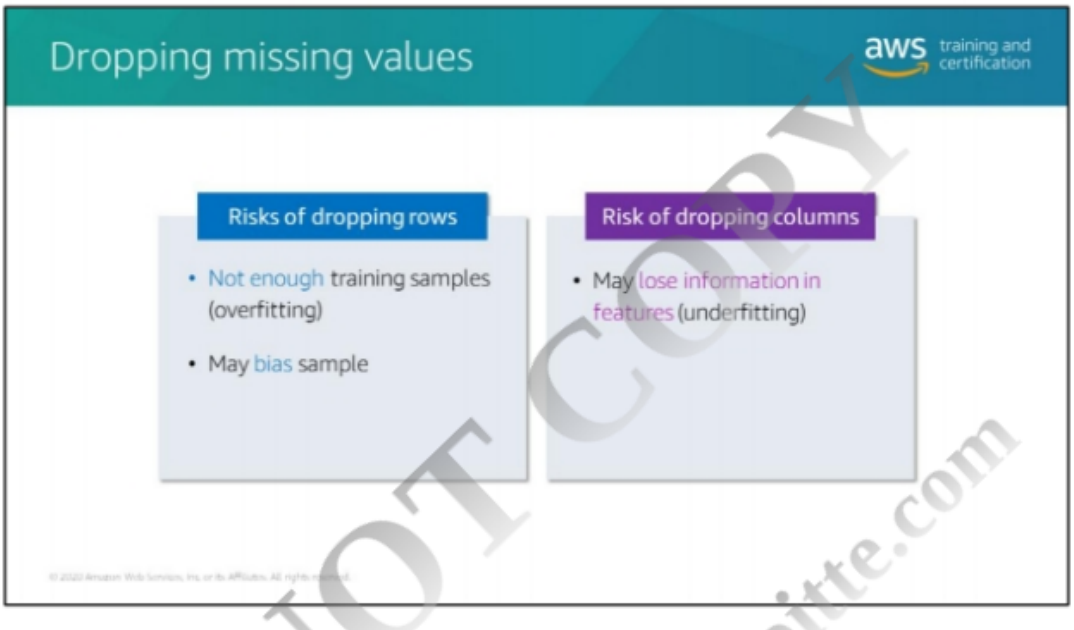
	Education	State
0	4 yr degree	New York
1	Baccalauréat	Minnesota
3	HS diploma	California

	Education
0	4 yr degree
1	Baccalauréat
2	Masters/PhD
3	HS diploma
4	Masters/PhD

Once you identify the missing values, the next step is determining how you're going to deal with them. One of the simplest ways – when looking at missing values within rows – is to just remove the row using the "dropna" function from Pandas.

If we do not give "dropna" any other parameters it is going to drop any and all rows with missing values. As we see, now after the missing value rows are dropped we only have three observations left. The "dropna" function can also be applied to columns, with the "axis=1" keywords.

We can get more sophisticated with the "dropna" function by setting up different thresholds and searching only within a particular subset of your data.



The risk of dropping rows is pretty significant. If you have many columns and each column has certain rows with missing values, you can see that if you drop all of them there's not many rows left. So, you may lose too much data, and once you dropped all those rows with missing values the data left may be biased.

Dropping columns has similar effects. You may have a few hundred columns, but maybe the majority of the columns are going to have certain missing values. If you're using "dropna" and there's only one observation missing in a particular column, then the entire column will be dropped. When you drop too many columns you may not have enough features to feed the model.

Imputing missing numerical data with Scikit-learn

aws training and certification

```
from sklearn.impute import SimpleImputer
import numpy as np

arr = np.array([[5,3,2,2],[3,None,1,9],[5,2,7,None]])

imputer = SimpleImputer(strategy='mean')
imp = imputer.fit(arr)
imputer.transform(arr)
array ([[ 5. , 3. , 2. , 2. ],[ 3. , 2.5, 1. , 9. ],[ 5. , 2. , 7. , 5.5]])
```


As an alternative to dropping missing values, you can imput values for those missing values.

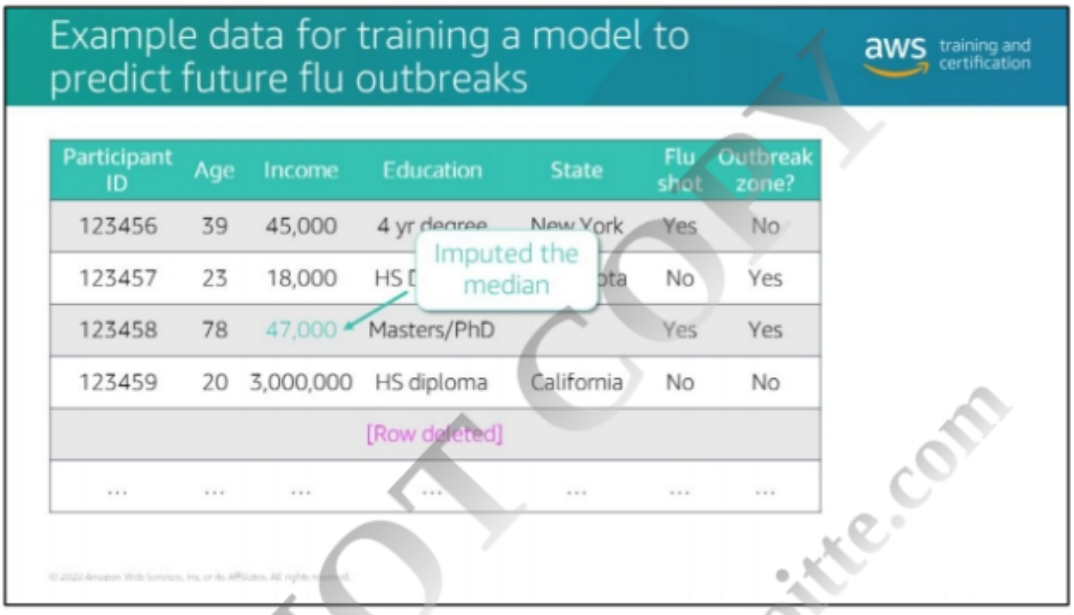
There are different ways to impute a missing value. Usually we replace the missing value with the mean, median, or most frequent values for categorical variables, and mean or median for numerical or continuous variables.

Let's look at an example. Here, we're using the Scikit-Learn imputer function for us to impute some missing values. It is a fairly small dataset, but we do have two missing values.

And the missing value we imputed here is by the strategy of mean. We calculate the mean - the mean of three and two is two point five - and imput the value for the missing value.

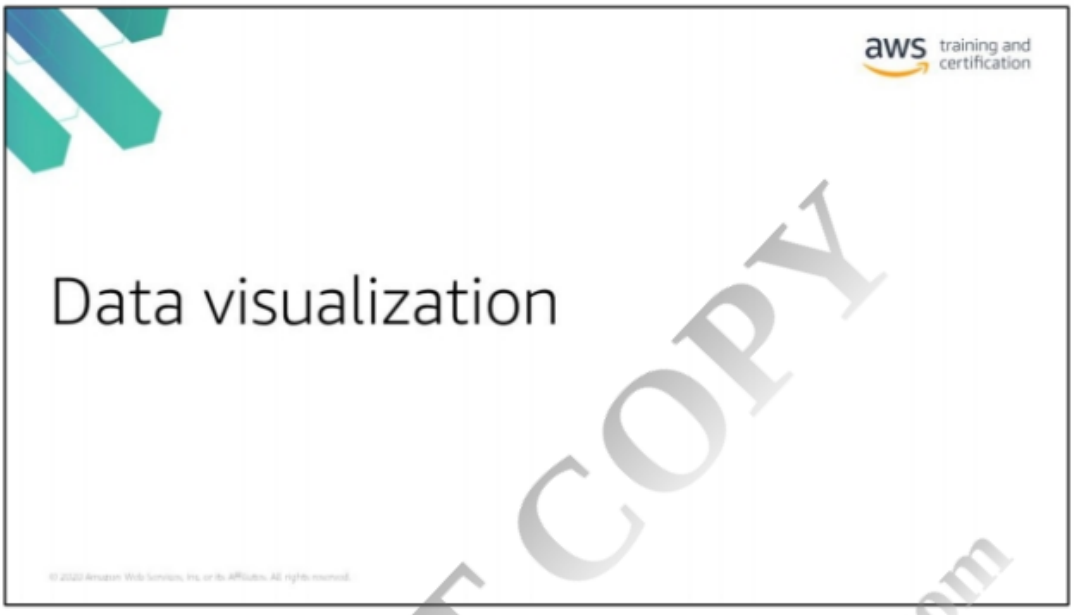
There are more advanced methods for imputing missing values, such as the multiple imputation by chained equations, which is available in the newer version of the Scikit-Learn.

In Python there is a fancy impute package which provides K nearest neighbor, soft impute, multiple imputation by chain equations, and a few other more advance imputation methodologies.



So to handle the dirty data, we converted the income column data to their natural logs. This reduces the numeric variance between the values, making it so the highest values aren't skewing the data. We didn't have my income value for Participant 123458, so we used the median value for that column and then converted it to its natural log.

Some of the empty values were left empty, since they were categorical and could not be replaced with mean, median, mode, or logarithmic values. Those rows were deemed good enough to keep, however, as they still had mostly useful data in them.

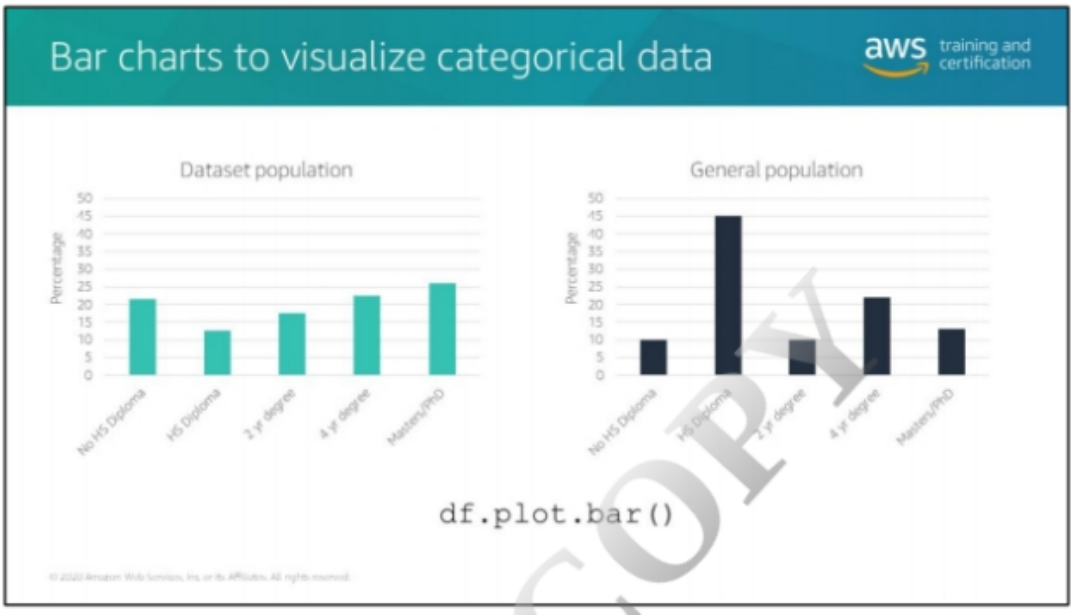


Visualization for categorical data

aws training and certification

Participant ID	Age	Income	Education	State	Flu shot	Outbreak zone?
123456	39	45,000	4 yr degree	New York	Yes	No
123457	23	18,000	HS Diploma	Minnesota	No	Yes
123458	78	47,000	Masters/PhD		Yes	Yes
123459	20	3,000,000	HS diploma	California	No	No
...	...	...	...	...	...	...

Let's go back to the descriptive stats table, and talk about how you can visualize different features and stats based on whether they're numerical or categorical.



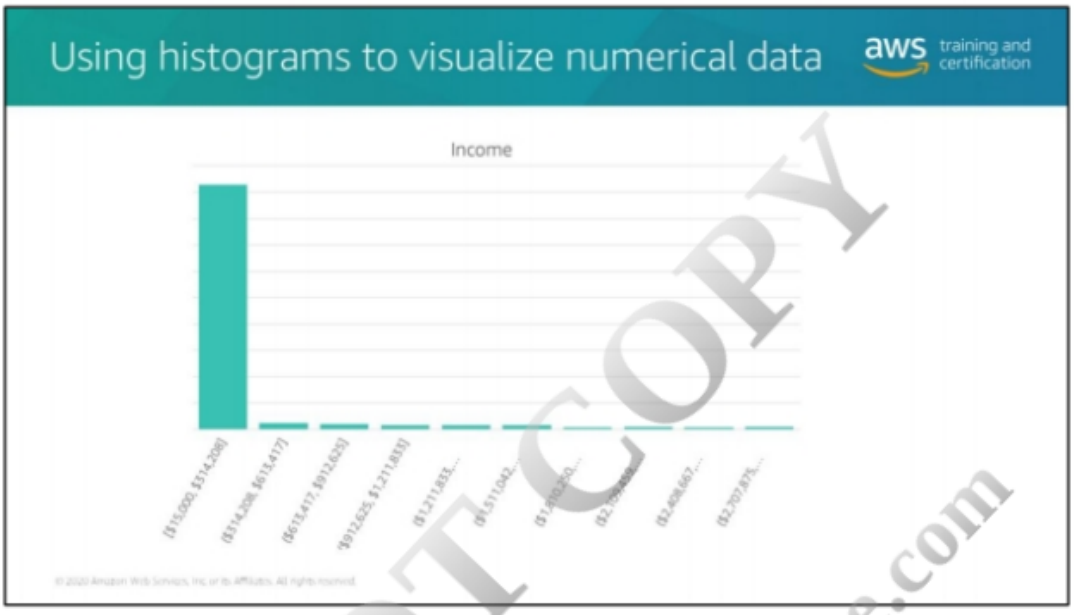
By visualizing the education data in our dataset and comparing it with the general population, we can tell that our data is strongly under-representing people with a HS diploma and over-representing people with no HS diploma, a 2-year degree, and those with a Masters or PhD. This means you'll need to keep this in mind as you evaluate the performance of the model. You may even want to ask your data scientist to perform some transformations on your data to help compensate for these problems.

Visualization for numerical data

aws training and certification

Participant ID	Age	Income	Education	State	Flu shot	Outbreak zone?
123456	39	45,000	4 yr degree	New York	Yes	No
123457	23	18,000	HS Diploma	Minnesota	No	Yes
123458	78	47,000	Masters/PhD		Yes	Yes
123459	20	3,000,000	HS diploma	California	No	No
...	...	...	...	...	...	...

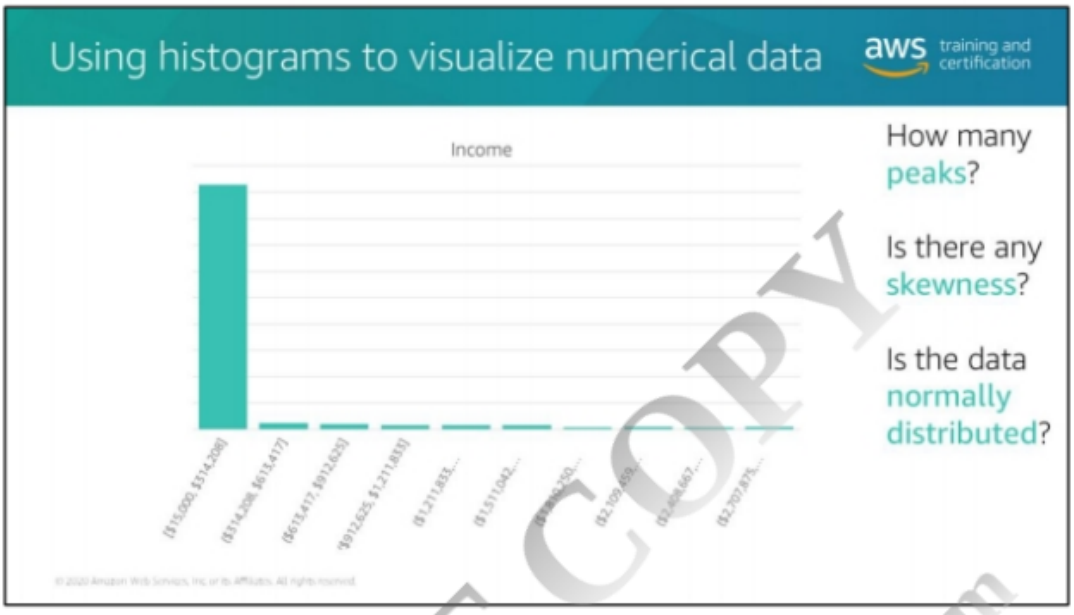
Here we see another interesting part of the dataset, which is that there's a huge gap between the mean and median incomes in our dataset. This implies that a small portion of the participants, those with the highest income, are strongly weighting the average income of the dataset upwards.



A histogram is often a good visualization technique to use in order to see the overall behavior of a particular feature. With a histogram, you can answer questions like these:

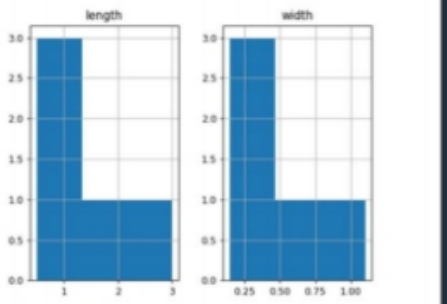
- Is the feature data normally distributed?
- How many peaks are there in the data?
- Is there any skewness for that particular feature?

When using histograms for your data visualization, values are binned. Binning is a technique we'll come back to in a later module on feature engineering. The taller peaks of the histogram indicate the most common values.



Create histograms from Pandas DataFrames

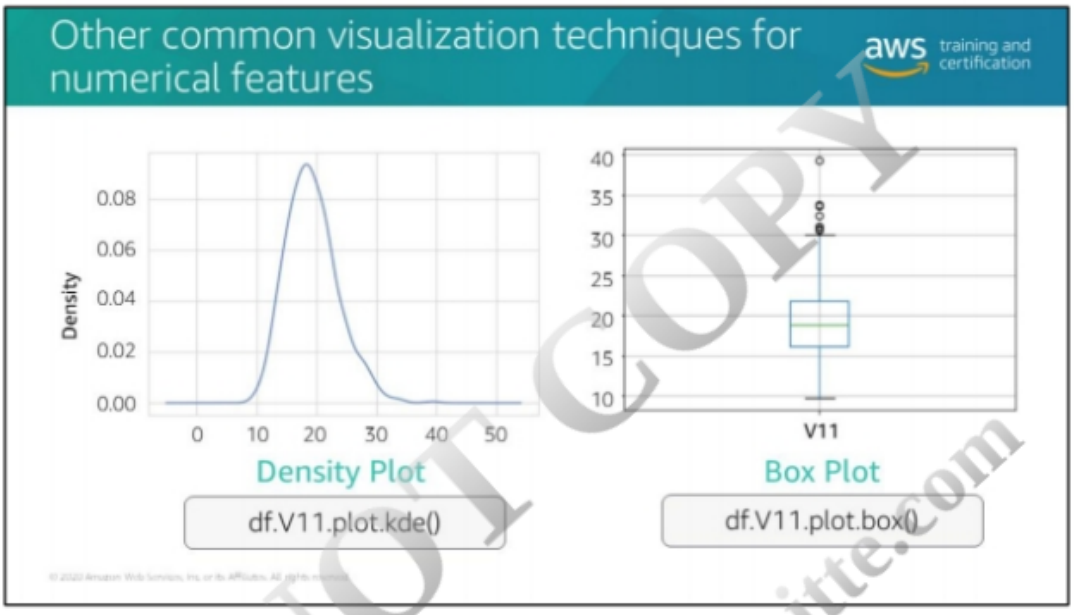
aws training and certification




```
In [10] : df = pd.DataFrame({
            'length': [1.5, 0.5, 1.2, 0.9, 3],
             'width': [0.7, 0.2, 0.15, 0.2, 1.1]
            },
            index= ['pig', 'rabbit', 'duck', 'chicken', 'horse'])

            hist = df.hist(bins=3)
```

You can create a histogram by using the Pandas 'hist' function, or Seaborn's 'distplot' function.

For more information, see https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.hist.html.

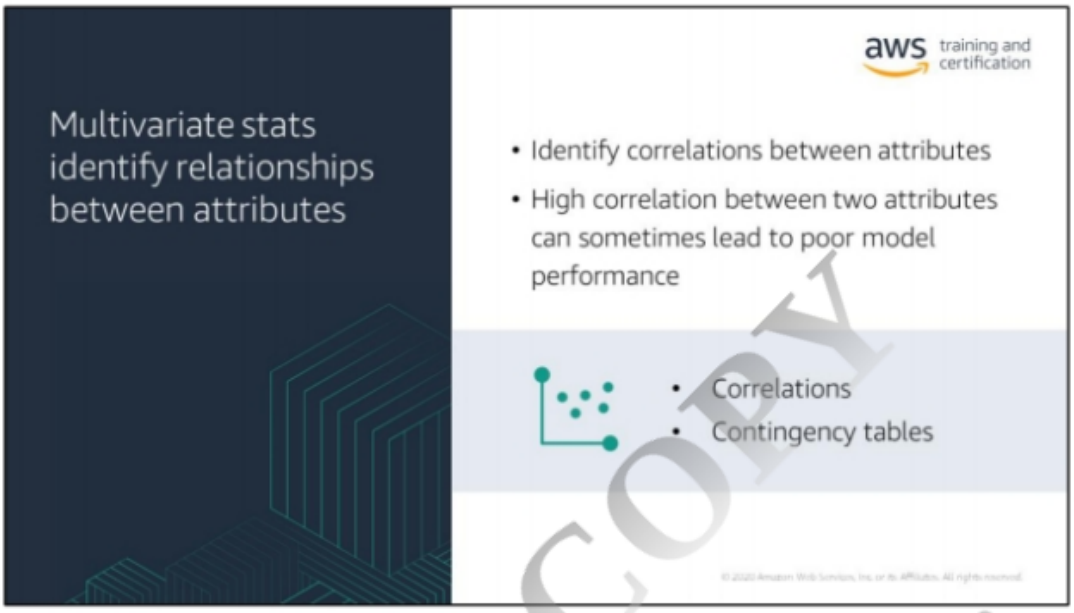


For numerical features, in addition to histograms, you can use density plots and box plots to get an idea of what's inside that particular feature. Like a histogram, these visualizations will help you answer questions like these:

- What's the range of the data? The peak of the data?
- Are there any outliers?
- Are there any special features?

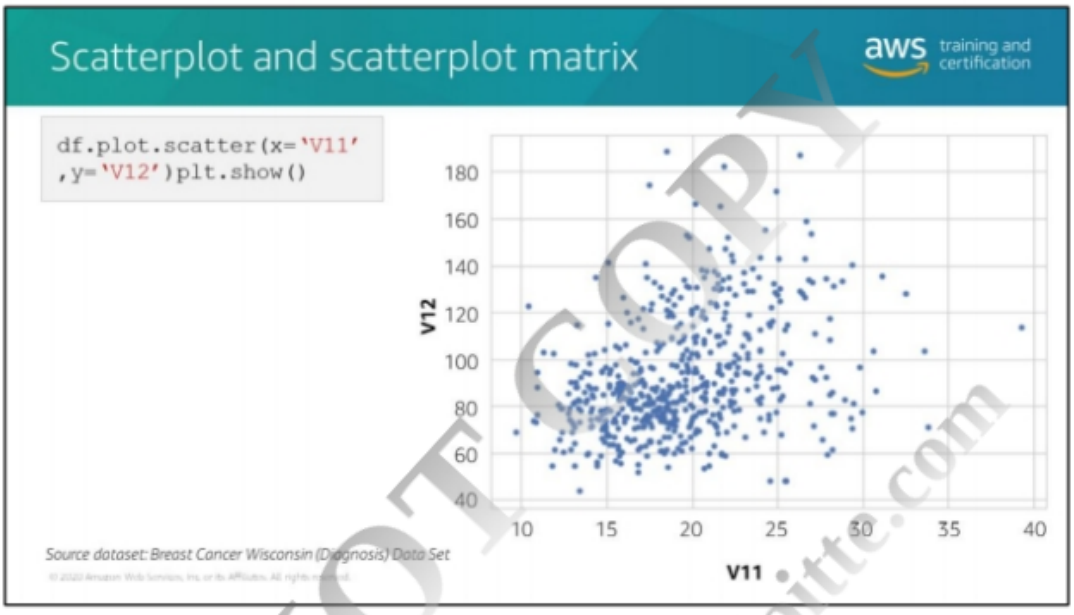
Answering these questions not only helps you understand your data better but also can help you decide if you need to do some more specialized data preprocessing. The *density plot* plots the distribution of your single feature. The density plot is similar to a histogram but plots a smooth version of the histogram density using a kernel density function. Another way of visualizing the distribution is a Box plot. A Box plot uses the Interquartile range to plot the distribution of your feature.

For more information, see https://en.wikipedia.org/wiki/Interquartile_range.



For looking at relationship between more than one variable, you can look at what are considered *multivariate statistics*. This mostly has to do with the correlations and relationships between your attributes.

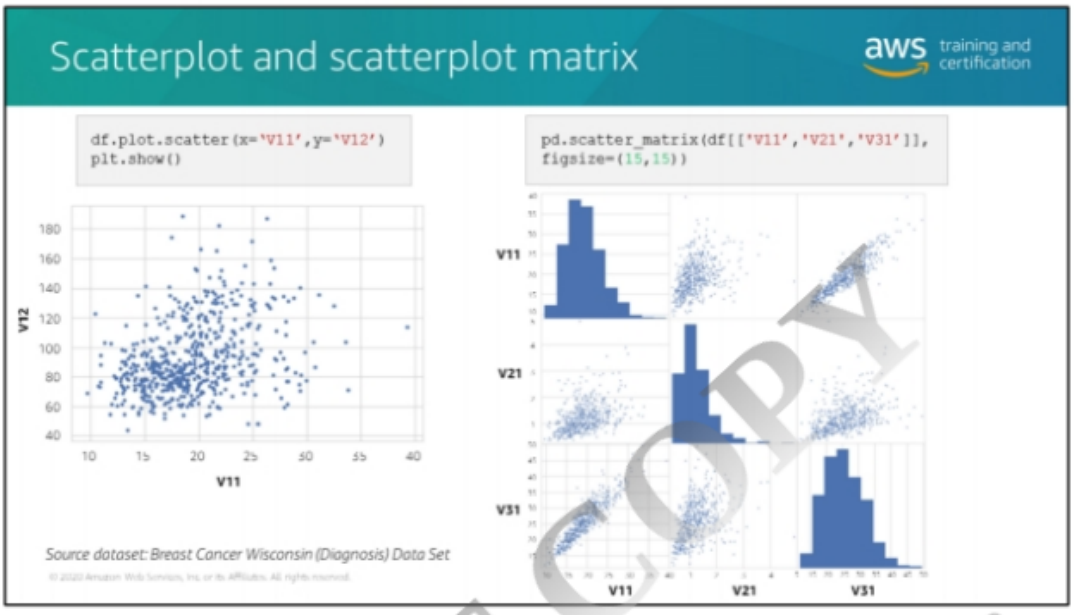
For cases when you have multiple variables or features, you may want to look at the correlations between them. It's important to identify correlations between attributes because high correlation between two attributes can sometimes lead to poor model performance. When features are closely correlated and they're all used in the same model to predict the response variable, there could be problems—for example, like the model loss not converging to a minimum state. So be aware of highly correlated features in your dataset.



When we have more than two numerical variables in a feature dataset, sometimes we want to look at their relationship and a scatterplot is a really good way to spot any special relationships among those variables.

For example, in this case, we have V11 and V12. Those are two numerical variables and we want to show their relationship so we're using a scatterplot to help us visualize that. As you can see, there are plots scattered around, and even though the correlation among them may not be that high because the data is scattered around quite a lot, there may be some relatively positive relationships between the two variables.

Source dataset:
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)



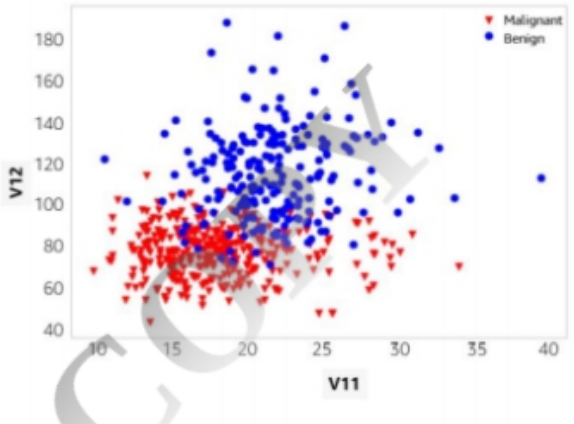
Scatterplot matrices help you look at the relationship between multiple different features.

In Pandas, we can easily create scatterplot matrices based on the columns we would like to look at it. In this case, we have three columns and it's going to give us the pair-wise scatterplot for any two of them.

Scatterplot with identification

aws training and certification

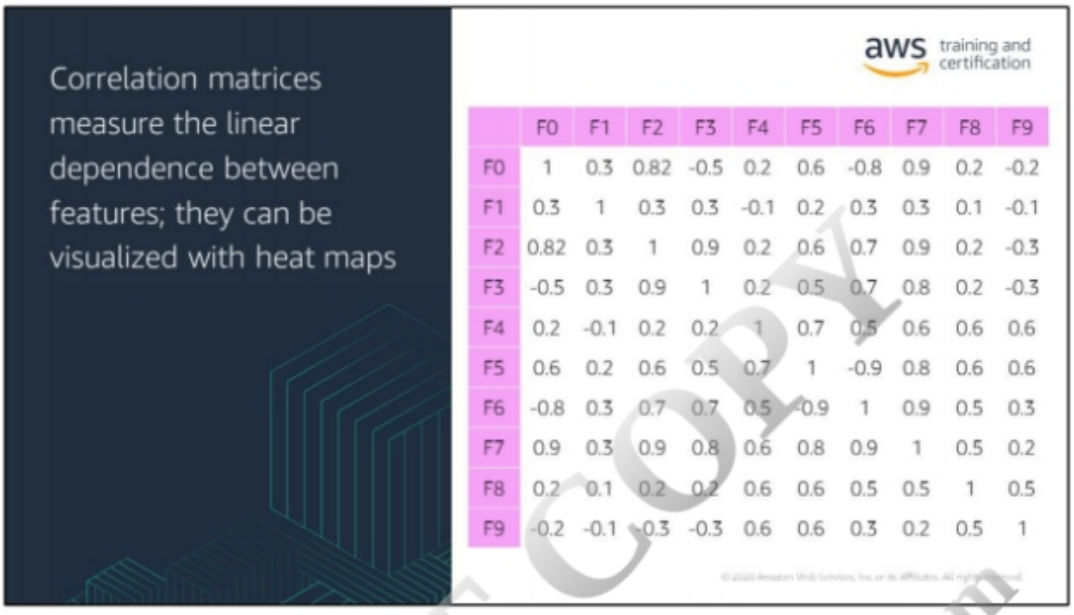
```
malignant = df [ ['V11', 'V12'] ] [ df ['target'] == 0]
benign =  df [ ['V11', 'V12'] ] [ df ['target'] == 1]

plt.scatter (benign ['V11'],
             (benign ['V12'],
              s=50,
              c='red',
              marker='v',
              label='Malignant')

plt.scatter (malignant ['V11'],
plt.scatter (malignant ['V12'],
              s=50,
              c='blue',
              marker='o',
              label='Benign')
plt. xlabel ('V11')
plt. ylabel ('V12')
plt. legend ()
plt.grid ()
plt.show ()
```



Source dataset: Breast Cancer Wisconsin (Diagnosis) Data Set

With a scatterplot, you may sometimes want to identify special regions that a particular subset of data may fit into. In this breast cancer data example, we have malignant as well as benign patients and here we highlight the difference between those two sub-groups by different colors and symbols.

By looking at V11 and V12, again, we see there are some boundaries between those two groups and even though they cannot be separated perfectly, we can see that there is indeed some power, such that those two variables, V11 and V12, can be used to distinguish the malignant or benign patients. This gives us an idea of how useful particular variables can be if we are using them for a classification problem.



	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
F0	1	0.3	0.82	-0.5	0.2	0.6	-0.8	0.9	0.2	-0.2
F1	0.3	1	0.3	0.3	-0.1	0.2	0.3	0.3	0.1	-0.1
F2	0.82	0.3	1	0.9	0.2	0.6	0.7	0.9	0.2	-0.3
F3	-0.5	0.3	0.9	1	0.2	0.5	0.7	0.8	0.2	-0.3
F4	0.2	-0.1	0.2	0.2	1	0.7	0.5	0.6	0.6	0.6
F5	0.6	0.2	0.6	0.5	0.7	1	-0.9	0.8	0.6	0.6
F6	-0.8	0.3	0.7	0.7	0.5	-0.9	1	0.9	0.5	0.3
F7	0.9	0.3	0.9	0.8	0.6	0.8	0.9	1	0.5	0.2
F8	0.2	0.1	0.2	0.2	0.6	0.6	0.5	0.5	1	0.5
F9	-0.2	-0.1	-0.3	-0.3	0.6	0.6	0.3	0.2	0.5	1

But how can you quantify the linear relationship among the variables you're seeing in a scatterplot? A correlation matrix is a good tool in this situation, because it conveys both the strong and weak linear relationships among numerical variables.

For correlation, it can go as high as one, or as low as minus one. When the correlation is one, this means those two numerical features are perfectly correlated with each other. It's like saying Y is proportional to X. When those two variables' correlation is minus one, it's like saying that Y is proportional to minus X. Any linear relationship in between can be quantified by the correlation. So if the correlation is zero, this means there's no linear relationship—but it does *not* mean that there's no relationship. It's just an indication that there is no linear relationship between those two variables.

Correlation matrix heat map

aws training and certification

```
col = ['V10', 'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18',
'V19']
hcatmap = np.corrcoef (df[col].values.T)

fig, ax = plt.subplots (figsize=(15,15))
im = ax.imshow (hcatmap, cmap='PIYG', vmin=-1)
fix.colorbar (im)
ax.grid (False)
[[ax.text (j, i, round (hcatmap [i, j], 2), ha="center", va="center",
color="w")
   for j in range (len (hcatmap)) ] for i in range (len (hcatmap)) ]

ax.set_xticks (np.arange (len (col)) )
ax.set_yticks (np.arange (len (col)) )
ax.set_xticklabels (col)
ax.set_yticklabels (col)

plt.show()
```

	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
F0	1	0.3	1	1	0.2	0.6	-0.8	0.9	0.2	-0.2
F1	0.6	1	0.3	0.3	-0	0.2	0.3	0.3	0.1	-0.1
F2	1	0.3	1	0.9	0.2	0.6	0.7	0.9	0.2	-0.3
F3	1	0.3	0.9	1	0.2	0.5	0.7	0.8	0.2	-0.3
F4	0.2	-0	0.2	0.2	1	0.7	0.5	0.6	0.6	0.6
F5	0.6	0.2	0.6	0.5	0.7	1	-0.9	0.8	0.6	0.6
F6	-0.8	0.3	0.7	0.7	0.5	-0.9	1	0.9	0.5	0.3
F7	0.9	0.3	0.9	0.8	0.6	0.8	0.9	1	0.5	0.2
F8	0.2	0.1	0.2	0.2	0.6	0.6	0.6	0.5	1	0.5
F9	-0.2	-0.1	-0.3	-0.3	0.6	0.6	0.5	0.2	0.5	1

However, looking at a number is not always straightforward. Often it's easier to view the numbers represented by colors. Now let's look at a heatmap. We have the highest number - one - in green and minus one in purple. The color actually gives us both the positive and negative directions, as well as how strong the correlations are.

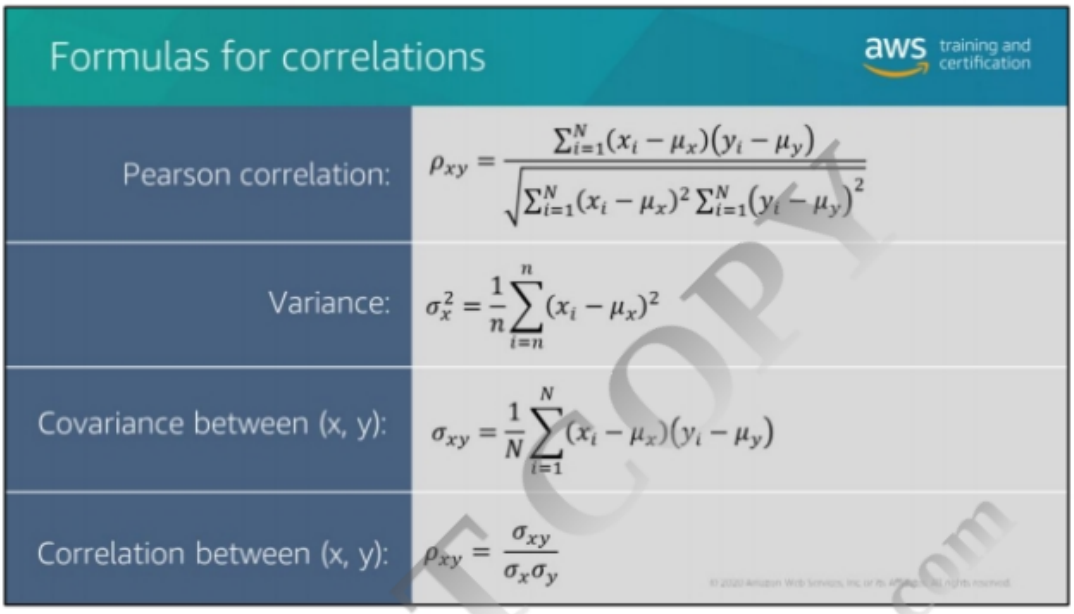
Correlation matrix heat map

aws training and certification

```
import seaborn as sns
sns.heatmap (heatmap, yticklabels=col, xticklabels=col, cmap='PiYG', annot=True)
```

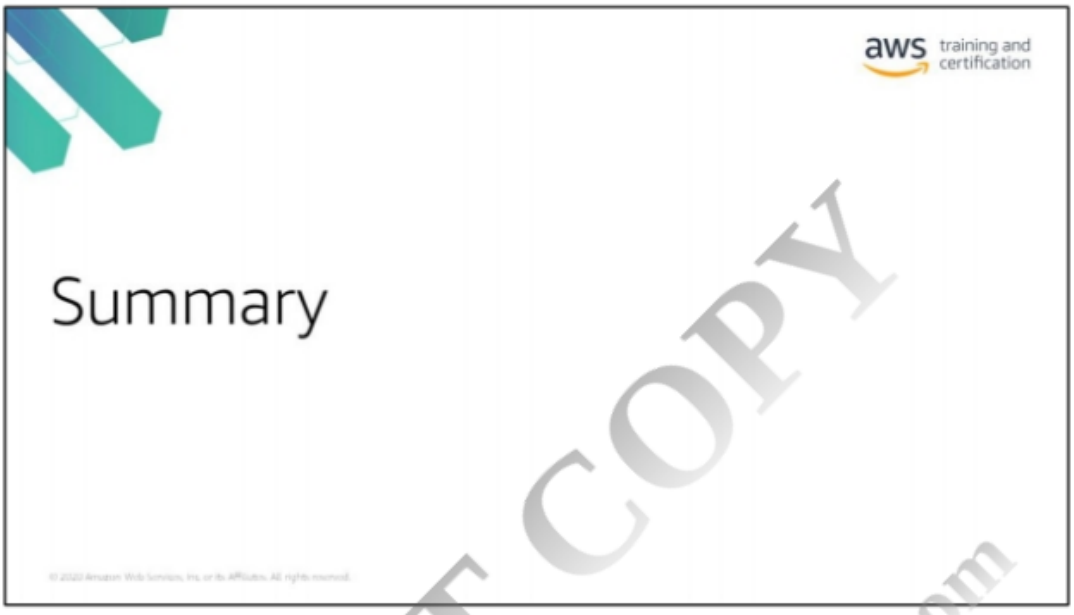
	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
F0	1	0.3	1	1	0.2	0.6	-0.8	0.9	0.2	-0.2
F1	0.6	1	0.3	0.3	-0	0.2	0.3	0.3	0.1	-0.1
F2	1	0.3	1	0.9	0.2	0.6	0.7	0.9	0.2	-0.3
F3	1	0.3	0.9	1	0.2	0.5	0.7	0.8	0.2	-0.3
F4	0.2	-0	0.2	0.2	1	0.7	0.5	0.6	0.6	0.6
F5	0.6	0.2	0.6	0.5	0.7	1	-0.9	0.8	0.6	0.6
F6	-0.8	0.3	0.7	0.7	0.5	-0.9	1	0.9	0.5	0.3
F7	0.9	0.3	0.9	0.8	0.6	0.8	0.9	1	0.5	0.2
F8	0.2	0.1	0.2	0.2	0.6	0.6	0.6	0.5	1	0.5
F9	-0.2	-0.1	-0.3	-0.3	0.6	0.6	0.5	0.2	0.5	1

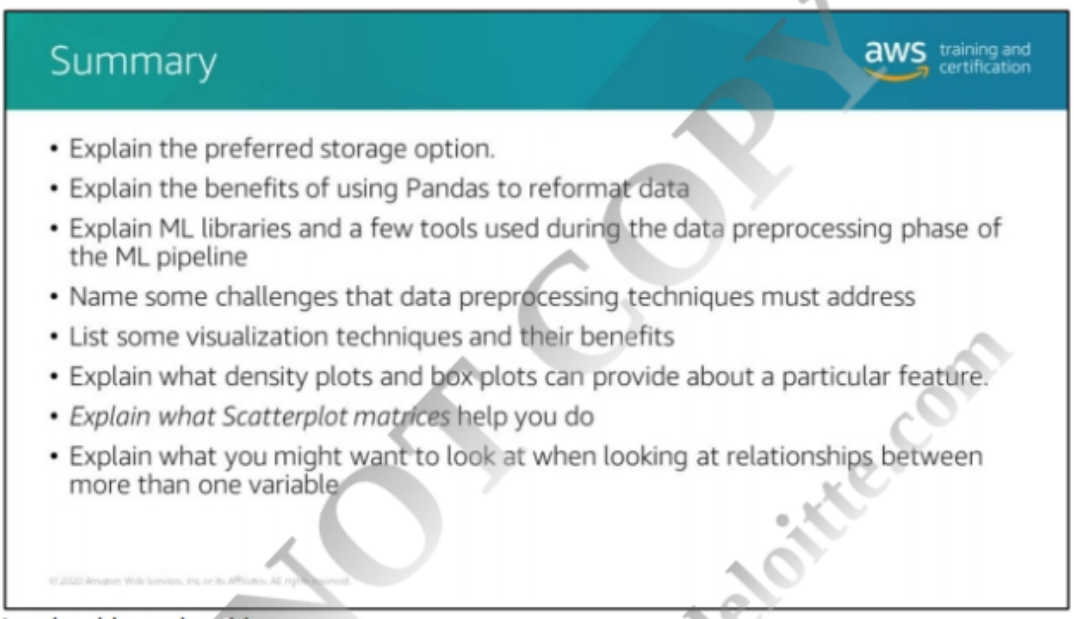
We can use the Seaborn's heat map function to show the correlation matrix.



How do you compute a correlation? If we look at the Pearson Correlation Equation, we can see there are a few components. First, on the top here is the covariance between the two variables, X and Y. And on the bottom, we have the standard deviation for each individual, X and Y.

Standard deviation is the square root of the variance. As you can see here, the square root of the variance for X is on the top with the square root of variance for Y being on bottom. The Pearson Correlation is just a covariance divided by the standard deviation of X and Y, as illustrated here. This is how we calculate the correlation. It is the covariance, as well as the standard deviation for each individual variable.





You should now be able to:

Explain the preferred storage option.

- Amazon S3 is the preferred storage option with a data lake for data science processing on AWS. Amazon S3 provides highly durable storage and seamless integration with various data processing services and ML platforms on AWS. It can be used as "one source of truth" storage for most AWS ML services.

Explain the benefits of using Pandas to reformat data

- With your data in a Pandas dataframe, you can calculate statistics, clean your data, visualize it, and even store the cleaned and transformed data back into its original format (e.g., a CSV file).

Explain ML libraries and a few tools used during the data preprocessing phase of the ML pipeline

- In machine learning, libraries are tools containing a premade set of routines and functions that enable you to format different types of data, like numerical or image data.
    - NumPy – for objects with multi-dimensional arrays
    - Scikit-Learn – for data mining and data analysis with Python
    - Matplotlib - A visualization library for Python used for two-dimensional plots of NumPy arrays
    - Seaborn - Another visualization library for Python that is built on top of Matplotlib

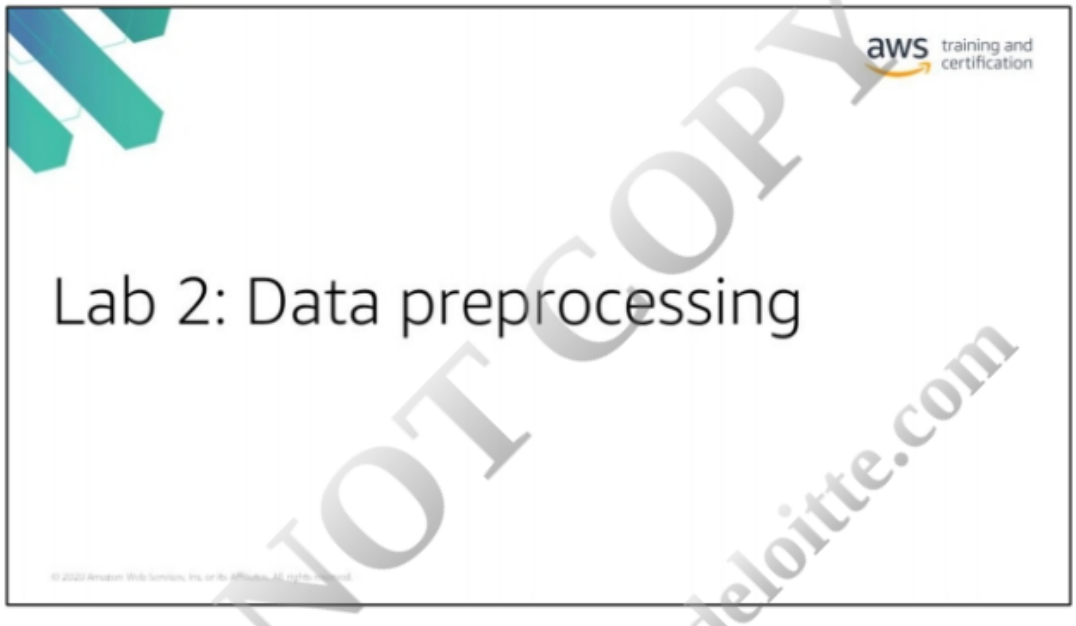
Name some challenges that data preprocessing techniques must address
- Missing data, outliers, Numerical and categorical data

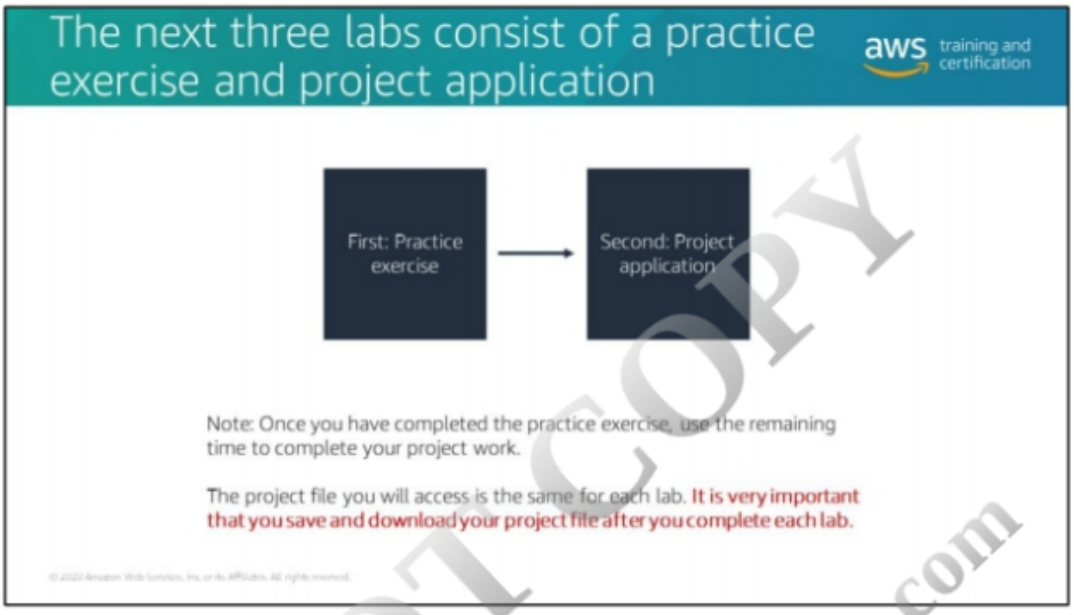
List some visualization techniques and their benefits
A histogram is often a good visualization technique to use in order to see the overall behavior of a particular feature. With a histogram, you can answer questions like these:
- Is the feature data normally distributed?
- How many peaks are there in the data?
- Is there any skewness for that particular feature?

Explain what density plots and box plots can provide about a particular feature.
- What's the range of the data? The peak of the data?
- Are there any outliers?
- Are there any special features?

Explain what Scatterplot matrices help you do
- They help you look at the relationship between multiple different features.

Explain what you might want to look at when looking at relationships between more than one variable
- You can look at what are considered *multivariate statistics*. This mostly has to do with the correlations and relationships between your attributes.

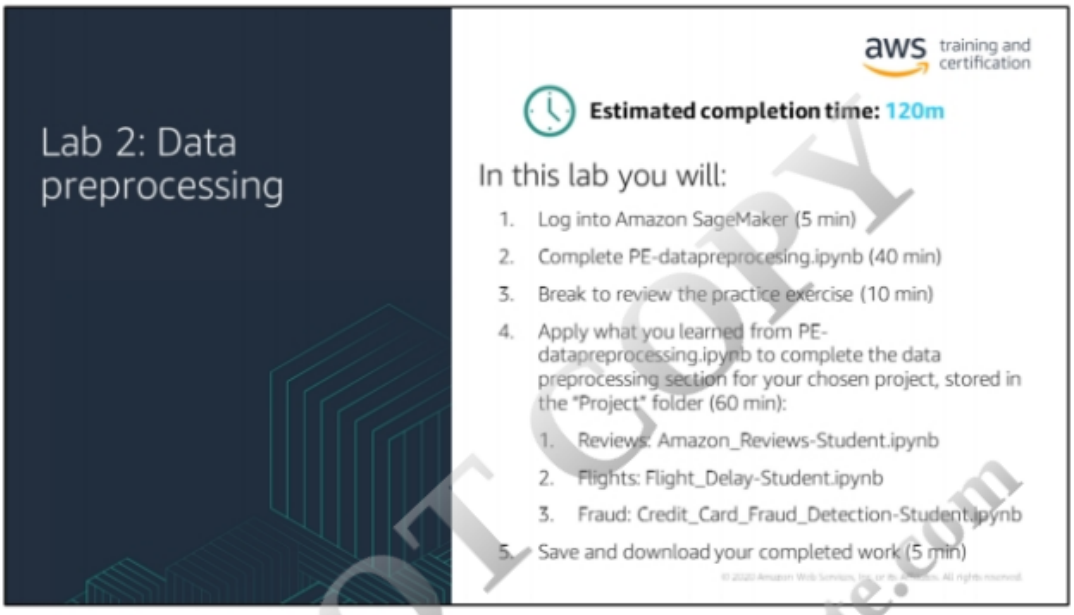




The first part of each lab (Task 1 in the lab instructions) is a practice exercise. This is where you will practice the skills related to the pipeline phase you're in. You will complete the exercise in the associated .ipynb that's provided in the instructions.

Once you are finished with the practice exercise, use the remaining time to complete your project work (Task 2 in the lab instructions). Follow the instructions to open the project notebook you chose and apply what you learned. The project file you will access is the same for each lab.

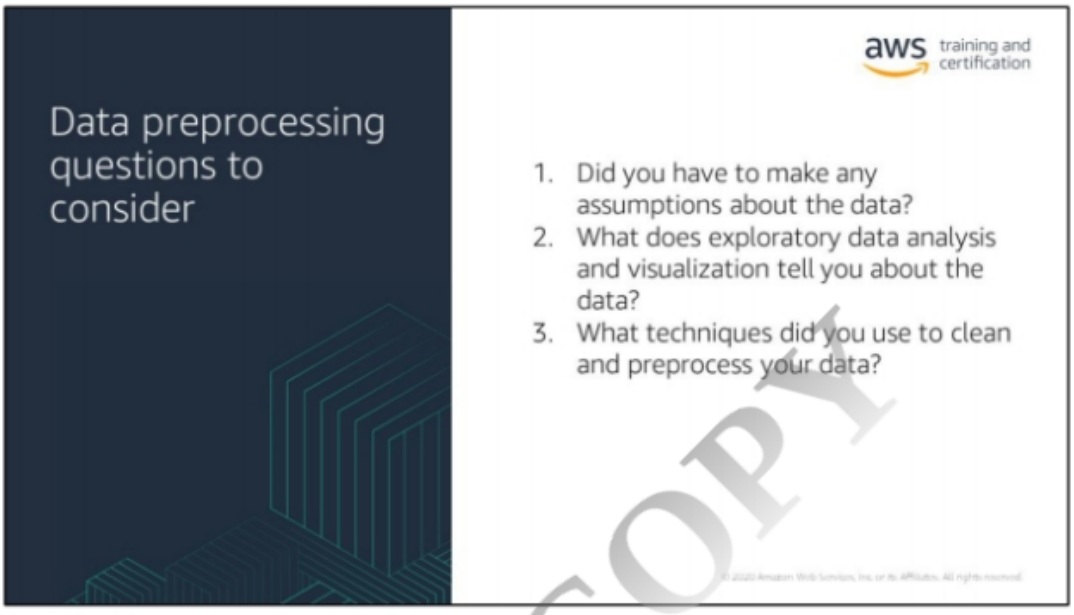
It is very important that you download your project file after you complete each lab. There are notes in the notebook indicating when you should pause to save and download.



Now let's transition to another exercise to walkthrough some of the concepts we've been talking about in the last couple of modules. In this exercise, we'll:

- Use descriptive statistics and data visualization techniques to better understand a dataset provided to you.
- Clean the data and deal with missing values and data outliers.

Note: Be sure to take the break to review the practice exercise.



As you work through data preprocessing for the business scenario you chose, be sure to track any relevant findings in your project template.